A stable decomposition algorithm for dynamical social network analysis

Romain Bourqui and Paolo Simonetto and Fabien Jourdan

Abstract Dynamic networks raise new knowledge discovery challenges. To handle efficiently this kind of data, an analysis method has to both decompose the network (modelled by a graph) into similar set of nodes and let the user detect structural changes in the graph. In this article we present a graph decomposition algorithm generating overlapping clusters. The complexity of this algorithm is $O(|E| \cdot deg_{max}^2 + |V| \cdot log(|V|)))$. This algorithm is particularly efficient due to its ability to detect major modifications along dynamic processes such as time related ones.

1 Introduction

Graph is a relevant data structure to organize large scale data; it is used in many application fields such as biology, micro-electronic or social sciences. Graph are particularly well suited in knowledge discovery since it exists a large range of algorithms to mine their structure and thus understand the underlying properties of the data (e.g. [Newman and Girvan(2004), Newman(2004), Palla et al(2007)Palla, Barabasi, and Vicsek, Suderman and Hallett(2007)]). In particular an intensive work is dedicated to the identification of clusters (communities) in these networks. For example, in social sciences, these algorithms detect individuals having the same field of interests. In biology, it helps in the identification of the proteins involved in same biological pro-

Romain Bourqui
Eindhoven University of Technology, P.O. Box 513; 5600 MB Eindhoven, Netherlands, e-mail: R.Bourqui@tue.nl

Paolo Simonetto
LaBRI, Université Bordeaux 1, 351, cours de la Libération F-33405 Talence cedex, e-mail: simonett@labri.fr

Fabien Jourdan
INRA, UMR1089, Xénobiotiques, F-31000 Toulouse, France, e-mail: Fabien.Jourdan@toulouse.inra.fr

cess (e.g. [Newman and Girvan(2004), Newman(2004), Palla et al(2007)Palla, Barabasi, and Vicsek, Bader and Hogue(2003)]). First purpose of these methods is to identify communities (clusters) but it also allows building visual abstraction of large network (Auber el al. [Auber et al(2003)Auber, Chiricota, Jourdan, and Melançon]).

Finding communities in a network is generally linked to a graph decomposition problem. Decomposition algorithms look for set of elements (clusters) sharing one or more properties. Generally, decomposition will be considered as a good one when elements within the sets are closely related while elements in different sets have low property coverage. In a more formal way: density within sets is larger than density between sets.

Dynamical data, in our case dynamical networks, are more and more present in datasets requiring knowledge discovery methods. In fact automatic data extractions are continuously improved (e.g. high throughput approaches in biology) and databases are populated quickly in biology (e.g. quantitative data on a organism evaluating according to environmental changes) or in social sciences (e.g. cocitation networks, movie actor networks). Consequently it is not only a question of identifying communities in a single snapshot of the network but also understanding the evolution of these communities along the different time frames. In other words it is necessary to be able to identify structural changes through, for instance, the spreading/collapsing of communities [Palla et al(2007)Palla, Barabasi, and Vicsek].

In this article we will apprehend the problem of dynamical network analysis by the use of graph decomposition. Our method first consists in dividing the dynamical process in a set of statistical snapshots of the network. Then, we apply a decomposition algorithm producing overlapping clusters for each of these static views of the network. Finally, in order to detect major changes during the dynamical process, we compare the decompositions using a similarity measure.

This article is organised as followed: in section 2, we present the overall approach, in section 3 the decomposition algorithm is described and finally, in section 4, we evaluate the stability of the decomposition on a social network.

2 Methodology

Figure 1 illustrates the method we propose in this article. The first step consists in turning the dynamical network into a set of static graphs. If we consider a dynamical graph G, defined on a time interval $[0..T]$, this transformation consists in building a set of static graphs $\{G_{[0,\varepsilon[}, ..., G_{[T-\varepsilon,T]}\}$, where ε is called the discretisation factor and $G_{[t,t+\varepsilon]}$ is the static graph corresponding tot the time period [t,t+] (i.e. this graph contains all the nodes and edges of the dynamic graph G being present during the period $[t, t+\varepsilon]$).

The leading concept of our approach is that if the graph changes a little (assuming that the discretisation factor is relevant) then two static graphs describing two consecutive periods have similar topological structures. To compare graph topologies, we first use a decomposition algorithm then we apply a graph decomposition

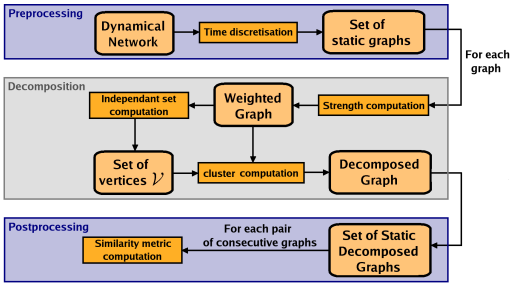

Fig. 1 Three main steps of our method.

similarity measure. This measure has to be low when comparing two graphs known has having a similar topology.

In this article we choose an algorithm which has the particularity to be stable when little topological changes appear between two networks. This algorithm is divided into three steps: firstly the strength metric [Auber et al(2003)Auber, Chiricota, Jourdan, and Melançon] is computed on edges and nodes, secondly we look for a maximal set of independent nodes (i.e. nodes at least at distance 2 from each other) and thirdly we build sets around these independent nodes.

3 Algorithm

3.1 Strength *metric*

To define Strength [Auber et al(2003)Auber, Chiricota, Jourdan, and Melançon, Chiricota et al(2003)Chiricota, Jourdan, we have to introduce some notations. Let u and v two nodes of graph G, we note $M_u(v) = N_G(v) \setminus (N_G(u) \cup \{u\})$ the set of neighbours of v (excluding u) that are not in the neighbourhood of u and we note $W_{uv} = N_G(u) \cap N_G(v)$ the set of nodes both neighbors of v and u. Let A and B be two set of nodes, we note $E(A,B)$ the set of edges linking a node in A and a node in B. Finally, $s(A,B) = |E(A,B)|/(|A| \cdot |B|)$ is the ratio between the number of edges linking A anb B and the maximum number of edges that could be linking these two sets [1]. Strength metric $e = (u,v)$ of an edge $w_s(e)$ is:

$$w_s(e) = \frac{\gamma_{3,4}(e)}{\gamma_{max}(e)} \tag{1}$$

Where:

$$\begin{aligned} \gamma_{3,4}(e) = |W_{uv}| + |E(M_v(u), M_u(v))| + |E(M_v(u), W_{uv})| \\ + |E(W_{uv}, M_u(v))| + |E(W_{uv})| \end{aligned} \tag{2}$$

[1] When $A = B$, then $s(A,A) = s(A) = 2 \cdot |E(A)|/(|A| \cdot (|A|-1))$

$$\begin{aligned}\gamma_{max}(e) = &|M_v(u)| + |W(u,v)| + |M_u(v)| + |M_v(u)||M_u(v)| \\ &+ |M_v(u)||W_{uv}| + |W_{uv}||M_u(v)| + |W_{uv}|(|W_{uv}|-1)/2\end{aligned} \tag{3}$$

This metric thus count, for each edge, the number of length 3 and 4 cycles going through this edge and then normalise this value regarding the maximum number of such cycles that can go through this edge.

Finally, we can define the Strenght metric on a node as follows:

$$w_s(u) = \frac{\sum_{e \in adj(u)} w_s(e)}{deg(u)}$$

Where $adj(u)$ is the set of edges adjacent to u et $deg(u)$ the degree of u. Thus Strength metric quantifies the connectivity ratio of the neighbourhood of an edge or a node. That is if a node or an edge is within a community its Strength value will be high. The time complexity to compute this metric is $O(|E| \cdot deg^2_{max})$ where deg_{max} is the maximum degree of the graph.

3.2 Extracting a maximal independent set

Our aim in this step is to find community centres that will be used to identify clusters. To do so we develop a method inspired of *MISF* (Maximal Independent Set Filtering) of Gajer and Kobourov [Gajer and Kobourov(2000)]. This approach consists in extracting a maximal set $\boldsymbol{V}$ of nodes such that $\forall u, v \in V$, $dist_G(u,v) \geq 2$. Selecting nodes at distance 2 in the graph allows obtaining a relevant node sampling of the network. Moreover the number of selected nodes also indicates the number of sets in the network. Finally this technique guaranties unicity of each group found by our algorithm (two sets can not contain exactly the same set of nodes).

Given that nodes in $\boldsymbol{V}$ will be centres of the communities, these nodes cant be considered as pivots in the network. In fact if we consider one of these pivots as a centre to compute a cluster, this cluster may contain several communities (see figure 2.(b)).

Strength metric computed on nodes allows identifying these pivots. In fact, nodes located at the intersection of several communities have a relatively low Strength value. Thus nodes with a strong Strength value have preferentially to be added to the set $\boldsymbol{V}$ (see figure 2.(c)). To select this set of nodes we developed the algorithm 1.

Sorting algorithm time complexity is $O(|V| \cdot log(|V|))$ and in space$O(|V|)$. Concerning **for** loop we can easily prove that its time and space complexity is $O(|V| + |E|)$. The overall cost of set $\boldsymbol{V}$ computation is in time $O(|V| \cdot log(|V|) + |E|)$ and in space $O(|V| + |E|)$.

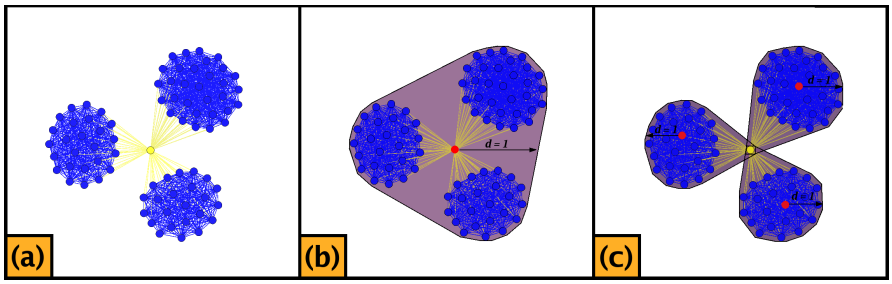

Fig. 2 (a) Subgraph of the "Hollywood graph" where vertices represent actors and two vertices are linked if the corresponding actors were involved in a common movie. Color of each vertex corresponds to its strength value, from the lowest in yellow to the highest in blue. (b) If the red vertex is choosen as a community "centre", we then could obtain one unique cluster containing the whole network. (c) If the community "centres" have high strength values, then we obtain 3 clusters corresponding to the 3 movies of that network.

Input: A graph $G = (V, E)$
Output: A maximal set $\mathcal{V}$ of vertices at distance at least 2
vector¡node¿ *sorted_nodes*;
sortNodeWithStrength(G, *sorted_nodes*);
for *unsigned int i from* 0 *to (number of vertices in G)* **do**
  node u = *sorted_nodes*[i];
  if u *in* G **then**
    append($\mathcal{V}$,u);
    foreach *node v in neighborhood of u* **do**
      remove(G, v);
    end
    remove(G, u);
  end
end

Algorithm 1: Extraction of the set $\mathcal{V}$. The sortNodeWithStrength(G, *sorted_nodes*) method sorts the vertices of G by decreasing Stength values and store the result in *sorted_nodes*.

3.3 Group detection

Alogrithm 2 then allows extracting clusters based on set $\mathcal{V}$. The idea of this algorithm is to build spheres of radius 1 in the graph around nodes of $\mathcal{V}$. For each node u in $\mathcal{V}$, if an edge (u, v) has a Strength value greater than a fixed threshold ε, then this edge has to be within the community of u. To compute the threshold we make the assumption that the less dense the network is, the less dense communities are. Thus threshold ε is computed according to the number of edges of graph $G = (V, E)$ and to the maximum number of edges of the complete graph $K_{|V|}$ with $|V|$ nodes. In algorithm 2, the threshold ε was fixed empirically and can be modified in order to build clusters more or less tolerant to noise.

Input: A graph $G = (V, E)$, the Strength of each edge, a maximal set $\mathcal{V}$
Output: A set D of groups of vertices
double $\varepsilon = 2 \cdot |E| / (|V| \cdot (|V| - 1))$;
foreach *node u in* $\mathcal{V}$ **do**
  Group *curGroup* = createNewGroup();
  append(*curGroup*, *u*);
  foreach *edge* $e = (u, v)$ *adjacent to u* **do**
    if $Strength(e) > \varepsilon$ **then**
      append(*curGroup*, *v*);
    end
  end
  append(*D*, *curGroup*);
end

Algorithm 2: Building groups of vertices.

For each node u in $\mathcal{V}$, algorithm 2 goes through edges adjacent to u and thus runs in a time complexity of $O(deg(u))$. The overall time complexity is $O(\sum_{u \in \mathcal{V}} deg_u)$. Given that $\sum_{u \in V} deg(u) = 2 \cdot |E|$ algorithm 2 has a time complexity of $O(|E|)$ and space complexity of$O(|V| + |E|)$.

Finally we can conclude that the overall complexity of the decomposition algorithm is in time $O(|E| \cdot deg_{max}^2 + |V| \cdot log(|V|))$ and in space $O(|V| + |E|)$.

4 Algorithm application

4.1 Case study

We chose as a case study a sub-part of the data set used in InfoVis 2007 Contest [InfoVis 2007 Contest(2007)]. This data set comes from a movie database IMDb (Internet Movie Database). In order to do this case study we extracted a subnetwork containing 432 movies and 4025 actors. We then built a graph as follows: a node in the graph is an actor and two nodes are connected by an edge if the corresponding actors were involved in at least one common movie. The resulting network contains 4025 nodes and 41216 edges. This network is presented on figure 3. This benchmark is particularly well suited for our study since it by essence contains communities. In fact, movies are cliques connecting all the actors who played in this movie, moreover cliques share nodes when an actor played in different movies.

In order to evaluate the quality of our decomposition algorithm, we use two criteria. First one is the quality of the result of the algorithm, that is the quality of the decomposition. To compute that quality, we use a generalisation of the measure MQ introduced by Mancoridis *et al.* [Mancoridis et al(1998)Mancoridis, Mitchell, Rorres, Chen, and Gansner]. This generalisation takes into account the cases where nodes can belong to several clusters. The second criteria measure the sensitivity of the algorithm to structural changes of the network during the dynamical process. To do so, we compare the

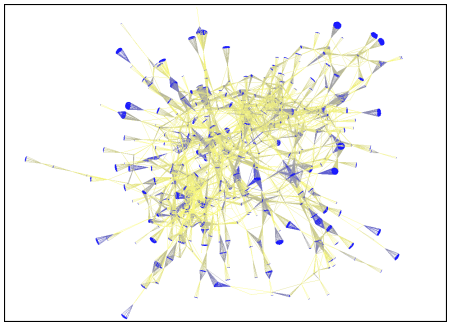

Fig. 3 Subgraph of the "Hollywood graph". This subgraph corresponds to a set of 432 movies and contains 4025 vertices (actors) and 41216 edges. Color of each vertex corresponds to its strength value (from the lowest values in yellow to the highest in blue).

decomposition obtained on the original graph to a decomposition obtained after several structural modifications of the network.

4.2 Decomposition quality

Criteria widely accepted to say that a set of clusters is a good graph decomposition is a high intra-cluster density and a low inter-cluster density. To compute that quality, we use a generalisation of the measure MQ introduced by Mancoridis *et al.* [Mancoridis et al(1998)Mancoridis, Mitchell, Rorres, Chen, and Gansner]. This method had been introduced by Bourqui and Auber [Bourqui and Auber(2008)]. Considering a graph $G = (V,E)$ and a decomposition $C = \{C_1, C_2, ..., C_k\}$ of nodes in G, MQ_{Over} is defined as follows:

$$MQ_{Over} = MQ^{+} - MQ^{-}_{Over} \tag{4}$$

Where

$$MQ^{+} = \frac{1}{k}\sum_{i} s(C_i, C_i) \tag{5}$$

And

$$MQ^{-}_{Over} = \frac{1}{k(k-1)}\sum_{i}\sum_{j\neq i} s_{Over}(C_i, C_j) \tag{6}$$

Where $s_{Over}(C_i, C_j) = \frac{|E(C_i, C_{j\setminus i})|}{|C_i|\cdot|C_{j\setminus i}|}$ et $C_{j\setminus i} = C_j \setminus (C_j \cap C_i)$. In that equation, MQ^{+} models the internal cohesion of clusters (intra-clusters) $C_1, ..., C_k$ while MQ^{-}_{Over} models the external cohesion of clusters (inter-clusters).

We applied our decomposition algorithm to the sub-network of the movie actor network (see Figure 3). Figure 4.(b) shows the result obtained on this graph. In this Figure, each cluster found by our algorithm is surrounded by a purple con-

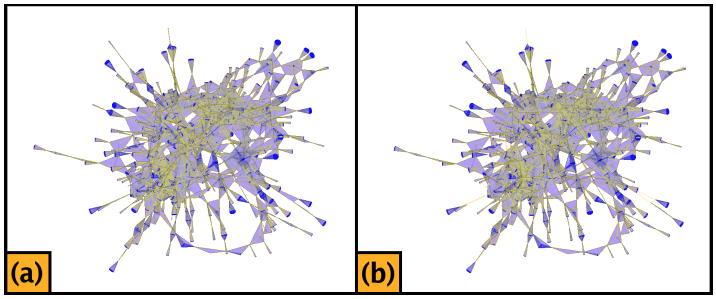

Fig. 4 Optimal decomposition (a) and the result of our algorithm (b) of the graph from Figure 3. Each group is surrounded by a purple convex hull.

vex hull. The value of MQ_{Over} is 0.95 showing that our algorithm gives excellent results according to this measure. Figure 4.(a) shows the optimal decomposition since it corresponds to the movies. We can first visually note the similarity of these two decompositions. When investigating into details the to results it appears that our algorithm finds 421 clusters among which 404 are perfectly fitting clusters of the optimal decomposition. Our algorithm thus found 93% of the optimal decomposition (and 96% of the clusters found correspond to movies). Regarding the 17 clusters obtained by our algorithm which are not fitting, each of this cluster is within a cluster of the optimal decomposition.

4.3 Sensitivity to modifications

In order to evaluate the sensitivity of our algorithm to network structural changes, we compare the reference decomposition (obtained on the original network) to the decompositions obtained after a given number of adding/removing of edges. In this section, we explain the similarity measure used to compare two decompositions. Then we present the result obtained in our case study.

4.3.1 Similarity measure

To measure the similarity of two decompositions, we use a metric inspired of the one proposed by Brohe and Van Helden [Brohée and van Helden(2006)]. This measure is based on *representativeness*. Two decompositions are considered as similar if and only if the first one is representative of the second and the opposite.

To define representativeness of two decompositions, we first have to define representativeness of two clusters. Let c_i and c_j two clusters, we say that the cluster c_i is representative of cluster c_j if and only if the cluster c_i is majorly composed of elements of cluster c_j. We define *directed cluster representativeness* as follows:

$$\rho_{c_i \to c_j} = \frac{c_i \cap c_j}{|c_j|} \qquad \rho_{c_j \to c_i} = \frac{c_i \cap c_j}{|c_i|}$$

We then define *undirected cluster representativeness* by:

$$\rho_{c_i c_j} = \sqrt{\rho_{c_i \to c_j} \cdot \rho_{c_j \to c_i}}$$

It corresponds to the geometrical mean of directed representativeness of clusters c_i and c_j.

We can, in a similar way, define the degree of representativeness of a decomposition regarding another one. Lets consider two decompositions C and C', we say that C is representative of C' if and only if for each cluster c' of decomposition C', the decomposition C contains a representative cluster of c'. Given that clusters of small size tends to bias this metric, we gives a higher representativeness to large size clusters. We define the *directed clustering representativeness* as follows:

$$\sigma_{C \to C'} = \frac{\sum_{c_i \in C'} \max_{c_j \in C} \rho_{c_j c_i} |c_i|}{\prod_{c_i \in C'} |c_i|}$$

This formula corresponds to the normalised weighted average of best representativeness of each cluster in C' by clusters in C.

We can then define the *undirected clustering representativeness* as follows:

$$\sigma_{CC'} = \sqrt{\sigma_{C \to C'} \cdot \sigma_{C' \to C}}$$

A possible modification of this metric could be obtained by using a simple product instead of the geometrical mean during the computations of the undirected representativeness. It allows distinguishing more easily similar decompositions from different ones. In fact, bad associations are then more penalised. In the next section we use this modified version of the similarity metric.

4.3.2 Experimental results

To measure the sensitivity of our decomposition algorithm, we first generate a dataset from graph showed on figure 3. To do so, we use algorithm 3. This algorithm allows to generate a collection of 100000 graphs.

We then compared the decomposition obtained on the original graph to those obtained on graphs from the sample *Collection* generated by algorithm 3. Figure 5 shows the results obtained.

On Figure 5.(a), the blue line shows the average number of perfect cluster matching according to the number of edges suppressed or added to the original network and the standard deviation is depicted in red for each of these average values. We can notice on this plot that up to 2000 operations, our algorithm allows finding in average between 250 and 421 (i.e. number of clusters in the original decomposition)

```
Input: subgraph G = (V,E) of the Hollywood graph
Output: A set Collection of graphs
for unsigned int i = 0 to i == NB_TESTS do
    Graph H = G;
    for unsigned int j = 0 to j == MAX_OPERATIONS do
        Operation op = getOperation();
        if op == 'edge deletion' then
            node src = getRandomNode();
            node tgt = getRandomNode();
            edge e = edge(src, tgt);
            while e is not element of H do
                src = getRandomNode();
                tgt = getRandomNode();
                e = edge(src, tgt);
            end
            deleteEdge(H, e);
        end
        else                                        /* op == 'edge addition' */
            node src = getRandomNode();
            node tgt = getRandomNode();
            edge e = edge(src, tgt);
            while e is element of H do
                src = getRandomNode();
                tgt = getRandomNode();
                e = edge(src, tgt);
            end
            addEdge(H, e);
        append(Collection, H);
    end
end
```

Algorithm 3: Generation of the dataset used to evaluate the stability of our decomposition algorithm. The getOperation() function returns 'edge addition' with a probability 0.5, 'edge deletion' otherwise. The constants NB_TESTS and $MAX\_OPERATIONS$ were respectively set to 50 and 2000.

perfect matching between original decomposition and decompositions obtained on sample graph collection. Moreover we can notice that the standard deviations are relatively low, between 0.44 and 10.34. On Figure 5.(b), the blue line shows the average value of the similarity metric according the number of edges removed or added to the original network and in red the standard deviations. Average values of the similarity metric stands between 0.9 and 1. This interval is very good in terms of similarity same for the standard deviations for which values are between 0.0002 and 0.007.

Considering the nave sensitivity measure which consists in computing the percentage of perfect matching, our algorithm preserves 78% of the clusters. Moreover average values of the similarity measure are also high showing the sensitivity of our decomposition algorithm.

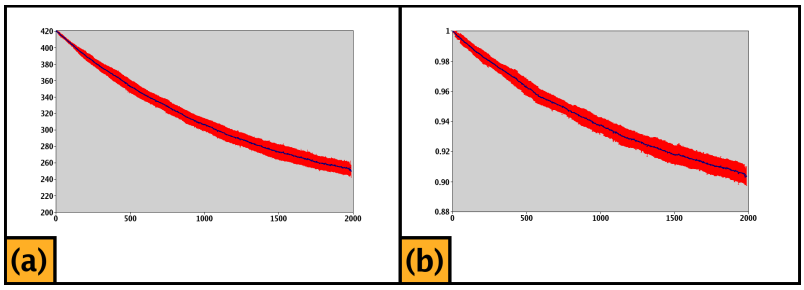

Fig. 5 (a) Average number of perfect matching (in blue) according to the number of addition or deletion operations previously done and the corresponding standard deviation (in red). (b) Average value of the similarity metric (in blue) according the number of addition or deletion operations previously done and the corresponding standard deviation (in red).

5 Conclusion

In this article, we present a new approach for the analysis of dynamic networks. This method is based on the transformation of the dynamic graph into a set of static graphs and on the graph decomposition in potentially overlapping clusters. Our main assumption is that is if the structure of the network doesnt evolve that much along the dynamical process, then decomposition obtained on two consecutive graphs should contain a similar community structure.

In order to decompose the graph corresponding to each time frame, we present a new graph decomposition algorithm of time complexity Given that in our approach to similar graphs should have similar decomposition, we show that our algorithm presents a low sensitivity to structural changes of the network.

Finally, we give a generalisation of [Brohée and van Helden(2006)] similarity measures to overlapping decompositions. It allows us to compare decomposition obtained of two graphs corresponding to two consecutive time frames and so to detect eventual high impact structural changes in the network.

References

[Auber et al(2003)Auber, Chiricota, Jourdan, and Melançon] Auber D, Chiricota Y, Jourdan F, Melançon G (2003) Multiscale Visualization of Small-World Networks. In: Proc. of IEEE Information Visualization Symposium, pp 75–81

[Bader and Hogue(2003)] Bader GD, Hogue CW (2003) An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinformatics 4

[Bourqui and Auber(2008)] Bourqui R, Auber D (2008) Analysis of 4-connected components decomposition for graph visualization. Tech. rep., LaBRI (http://www.labri.fr/)

[Brohée and van Helden(2006)] Brohée S, van Helden J (2006) Evaluation of clustering algorithms for protein-protein interaction networks. BMC Bioinformatics 7(488), http://www.biomedcentral.com/1471-2105/7/488

[Chiricota et al(2003)Chiricota, Jourdan, and Melançon] Chiricota Y, Jourdan F, Melançon G (2003) Software Components Capture using Graph Clustering. In: 11th IEEE Int. Workshop on Program Comprehension

[Gajer and Kobourov(2000)] Gajer P, Kobourov SG (2000) GRIP: Graph dRawing with Intelligent Placement. In: Proc. Graph Drawing 2000 (GD'00), pp 222–228

[InfoVis 2007 Contest(2007)] InfoVis 2007 Contest (2007) IEEE InfoVis 2007 Contest: InfoVis goes to the movies. Http://www.apl.jhu.edu/Misc/Visualization/

[Mancoridis et al(1998)Mancoridis, Mitchell, Rorres, Chen, and Gansner] Mancoridis S, Mitchell BS, Rorres C, Chen Y, Gansner ER (1998) Using Automatic Clustering to Produce High-Level System Organizations of Source Code. In: IEEE Proc. Int. Workshop on Program Understanding (IWPC'98), pp 45–53

[Newman(2004)] Newman MEJ (2004) Fast algorithm for detecting community structure in networks. Physical Review E 69:066,133

[Newman and Girvan(2004)] Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Physical Review E 69:026,113

[Palla et al(2007)Palla, Barabasi, and Vicsek] Palla G, Barabasi AL, Vicsek T (2007) Quantifying social group evolution. Nature 446:664–667

[Suderman and Hallett(2007)] Suderman M, Hallett M (2007) Tools for visually exploring biological networks. Bioinformatics Advanced Access p in press, DOI 10.1093/bioinformatics/btm401